Trabajo Práctico 0: Infraestructura básica

Fabrizio Cozza, *Padrón Nro. 97.402*
fabrizio.cozza@gmail.com

Kevin Cajachuán, *Padrón Nro. 98.725*
kevincajachuan@hotmail.com

Luciano Giannotti, *Padrón Nro. 97.215*
luciano_giannotti@hotmail.com.ar

1er. Cuatrimestre de 2018
66.20 Organización de Computadoras – Práctica Martes
Facultad de Ingeniería, Universidad de Buenos Aires

1. Objetivos

Este Trabajo Práctico tiene el fin de ayudarnos a familiarizarnos con las herramientas de software que utilizaremos posteriormente en otros trabajos, como es el emulador **gxemul** para correr programas sobre una maquina MIPS con el Sitema Operativo NetBSD.

2. Programa

El software de este trabajo está escrito en lenguaje C y permite dibujar **Julia Sets** o **Conjuntos de Julia** segun los parámetros que le pasamos por línea de comando. Estos parámetros son la región del plano complejo: delimitada por un centro, un ancho y un alto; una semilla que afectará el cálculo para cada pixel; la resolución y la salida ya sea por pantalla o por archivo. El formato a usar es PGM o *portable gray format*, que resulta útil para describir imágenes digitales en escala de grises.

3. Implementación

Una vez recibidos los parámetros, para dibujar el Julia Set el programa obtiene de cada píxel de la ventana a un punto en el plano complejo. A ese punto se lo eleva al cuadrado y le suma la semilla mencionada en la sección anterior. Esto se repite hasta que el valor absoluto del resultado sea menor a 2, en cuyo caso se toma la cantidad de iteraciones y se imprime en el archivo PGM, representando el nivel de blanco de ese piíxel.

Implementamos la clase **complex** para representar los numeros complejos.

También implementamos las siguientes funciones dentro del main.c para dibujar los conjuntos:

```
complex addComplexNumbers(complex a,complex b)
```

Suma dos números complejos.

```
complex sqrComplex(complex a)
```

Calcula la raíz cuadrada de un número complejo.

```
double absComplex(complex a)
```

Calcula el valor absoluto de un complejo.

```
int isValidNumber(char* arg)
```

Verifica que los valores ingresados para el ancho y el alto de la imagen sean válidos. De ser así devuelve 1, caso contrario devuelve 0.

```
int isValidRes(char* arg)
```

Verifica que el valor ingresado para la resolución sea válido. De ser así devuelve 1, caso contrario devuelve 0.

```
int isValidComplex(char* arg)
```

Verifica que el valor ingresado para el centro sea válido. De ser así devuelve 1, caso contrario devuelve 0.

```
int processImage(int resW, int resH,
                 complex pPos, complex seed,
                 double w, double h,
                 FILE* im, int N)
```

Contiene la lógica del dibujo de los sets. Recibe el alto y ancho de la misma, el centro, la semilla, la resolución y la cantidad de iteraciones máximas. Cada píxel dentro de la resolucíon ingresada es uno por uno transformado a números complejos teniendo en cuenta el ancho y alto y el centro y se procesa el nivel en la escala de grises correspondiente a ese píxel como se explica al principio de la sección. Estos niveles se escriben en un buffer y luego en un archivo siguiendo el formato PGM. La función retorna -1 si ocurrió algún error.

El programa se compila con el siguiente comando:

```
$gcc main.c -o tp0 -lm -std=gnu99
```

4. Código C

En esta sección colocaremos el código fuente del programa en lenguaje C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#define DEFAULT_WIDTH_RES 640;
#define DEFAULT_HEIGHT_RES 480;
#define DEFAULT_REALIMAGINARY 0;
#define DEFAULT_WIDTH_HEIGHT 2;
#define DEFAULT_REALSEED -0.726895347709114071439;
#define DEFAULT_IMAGINARYSEED 0.188887129043845954792;

typedef struct{
        double x,y;
}complex;

complex addComplexNumbers(complex a,complex b){
        complex c;
        c.x = a.x + b.x;
        c.y = a.y + b.y;
        return c;
}

complex sqrComplex(complex a){
        complex c;
        c.x = a.x*a.x - a.y*a.y;
        c.y = 2*a.x*a.y;
        return c;
}
```

```
double absComplex(complex a){
        return sqrt(a.x*a.x + a.y*a.y);
}

int processImage(int resW, int resH,
                 complex pPos, complex seed,
                 double w, double h,
                 FILE* im, int N){

        int x,y,i;
  complex z0,z1;

    if(fprintf(im, "P2 \n") < 0){
        perror("Error impreso por perror");
        return -1;
    }
    if(fprintf(im, "%d %d \n",resW,resH) < 0){
        perror("Error impreso por perror");
        return -1;
    }
    if(fprintf(im, "%d \n", N) < 0){
        perror("Error impreso por perror");
        return -1;
    }

  for(y=0;y<resH;y++){
        for(x=0;x<resW;x++){
                        // Valor z segun posicion del pixel
      z1.x=pPos.x-w/2+w/(double)resW/2+w/(double)resW*x;
      z1.y=pPos.y+h/2-h/(double)resH/2-h/(double)resH*y;
      z0.x = 0;
      z0.y = 0;
      for(i=0;i<N-1;i++){
                                z0 = addComplexNumbers(sqrComplex(z1),seed);
        z1=z0;
        if (absComplex(z0) > 2.0){
                break;
        }
        i++;
        }
      //agregar al buffer el brillo
      if(fprintf(im, "%3d ", i) < 0){
            perror("Error impreso por perror");
            return -1;
      }
    }
                if(fprintf(im, "\n") < 0){
             perror("Error impreso por perror");
             return -1;
                }
        }

        /* close the file */
        return fclose(im);
}

int isValidNumber(char* arg) {
        for(int i = 0; arg[i] != '\0'; i++) {
                if((arg[i] < '0') || (arg[i] > '9')) {
                        if(arg[i] == '.') {
                                if(i == 0 || arg[i + 1] == '\0') return 0;
                                continue;
                        }
                        return 0;
                }
        }
        return 1;
}

int isValidRes(char* arg) {
        int x = 0;
        for(int i = 0; arg[i] != '\0'; i++) {
                if(arg[i] < '0' || arg[i] > '9') {
                        if(arg[i] == 'x') {
```

```
                                if(i == 0 || arg[i+1] == '\0') return 0;
                                x = 1;
                                continue;
                        }
                        return 0;
                }
        }
        return x;
}

int isValidComplex(char* arg) {
        int j = 0;
        int sign = 0;
        for(int i = 0; arg[i] != '\0'; i++) {
                if(arg[i] < '0' || arg[i] > '9') {
                        if(arg[i] == '.') {
                                if(i == 0 || arg[i+1] == 'i' || arg[i+1] ==
                                     '+') return 0;
                                continue;
                        }

                        else if(arg[i] == 'i') {
                                if(arg[i+1] != '\0') return 0;
                                j = sign;
                                continue;
                        }

                        else if(arg[i] == '+') {
                                if(arg[i+1] == '-' || arg[i+1] == 'i' || arg[
                                     i+1] == '.' || i == 0) return 0;
                                sign = 1;
                                continue;
                        }

                        else if(arg[i] == '-') {
                                if(arg[i+1] == '+' || arg[i+1] == '.' || arg[
                                     i+1] == 'i') return 0;
                                if(i != 0) sign = 1;
                                continue;
                        }
                        return 0;
                }
        }
        return j;
}

int main(int argc, char* argv[]){
        int exitCode = 0;
        int pasoN = 500;

  int resWidth;
  int resHeight;
  complex pixelPos;
  double width;
  double height;
  complex seed;
        FILE* image = stdout;

  const char* delRes = "x";
        const char* delimitator = "+-i";
        char *pSeparator;

  resWidth = DEFAULT_WIDTH_RES;
  resHeight = DEFAULT_HEIGHT_RES;
  pixelPos.x = DEFAULT_REALIMAGINARY;
  pixelPos.y = DEFAULT_REALIMAGINARY;
  width = DEFAULT_WIDTH_HEIGHT;
  height = DEFAULT_WIDTH_HEIGHT;
  seed.x = DEFAULT_REALSEED;
  seed.y = DEFAULT_IMAGINARYSEED;

  for (int i = 1; i < argc; ++i){
        if ((((!strcmp(argv[i],"-V"))) || ((!strcmp(argv[i],"--version"))))){
        printf("TPO Organizacion de Computadoras version \"1.0.0\"\
```

```
            \n\nIntegrantes:\n Fabrizio Cozza\n Kevin CajachuÃ¡n\n Luciano
                Giannotti\n");
      return 0;
                }
    else if (((!strcmp(argv[i],"-h"))) || ((!strcmp(argv[i],"--help")))){
      printf("\
Uso:\n\
  tp0 -h\n\
  tp0 -V\n\
  tp0 [options]\n\
Opciones:\n\
  -V, --version     Version del programa.\n\
  -h, --help        Informacion acerca de los comandos.\n\
  -r, --resolution  Cambiar resolucion de la imagen.\n\
  -c, --center      Coordenadas correspondientes al punto central.\n\
  -w, --width       Especifica el ancho de la region del plano complejo por
       dibujar.\n\
  -H, --height      Especifica el alto de la region del plano complejo por
       dibujar.\n\
  -s, --seed        Configurar el valor complejo de la semilla usada para
       generar el fractal.\n\
  -o, --output      Colocar la imagen de salida.\n\
Ejemplos:\n\
  tp0 -o uno.pgm\n");
      return 0;
                }

    else if (!strcmp(argv[i], "-r") || !strcmp(argv[i], "--resolution")){
                        if(!argv[i+1] || !isValidRes(argv[i+1])){
        printf("Error: valor de resolucion ingresado no valido\n");
        return -1;
      } else {
        pSeparator = strtok(argv[i+1],delRes);
        if(pSeparator != NULL){
                resWidth = atof(pSeparator);
        } else {
                exitCode = -1;
                perror("Error impreso por perror");
        }

        pSeparator = strtok (NULL,delRes);
        if(pSeparator != NULL){
                resHeight = atof(pSeparator);
        } else {
                exitCode = -1;
                perror("Error impreso por perror");
        }
                                i++;
                        }
                }

                else if (!strcmp(argv[i], "-c") || !strcmp(argv[i], "--center
                     ")){
                        if(!argv[i+1] || !isValidComplex(argv[i+1])){
                                printf("Error: valor de centro ingresado no
                                     valido\n");
        return -1;
      } else {
        char *copy = strdup(argv[i+1]);
        if(copy == NULL){
                exitCode = -1;
                perror("Error impreso por perror");
        }
                                int sign = 1;
                                if(copy[0] == '-') sign = -1;

        pSeparator = strtok(argv[i+1],delimitator);
        if(pSeparator != NULL){
                pixelPos.x = sign * atof(pSeparator);
                                        int len = strlen(pSeparator);
                                        if(sign == -1) sign = copy[len + 1]
                                             == '-' ? -1 : 1;
                                        else sign = copy[len] == '-' ? -1 :
                                             1;
```

```
} else {
  exitCode = -1;
  perror("Error impreso por perror");
}

pSeparator = strtok (NULL,delimitator);
if(pSeparator != NULL){
        pixelPos.y = sign * atof(pSeparator);
} else {
  exitCode = -1;
  perror("Error impreso por perror");
}
                        free(copy);
                        i++;
        }
        }

        else if (!strcmp(argv[i], "-w") || !strcmp(argv[i], "--width
             ")){
                if(!argv[i+1] || !isValidNumber(argv[i+1])){
                        printf("Error: valor de ancho ingresado no
                             valido\n");
                        return -1;
                } else {
                        width = atof(argv[i+1]);
                        i++;
                }
        }

        else if (!strcmp(argv[i], "-H") || !strcmp(argv[i], "--height
             ")){
                if(!argv[i+1] || !isValidNumber(argv[i+1])){
                        printf("Error: valor de altura ingresado no
                             valido\n");
                        return -1;
                } else {
                        height = atof(argv[i+1]);
                        i++;
                }
        }

        else if (!strcmp(argv[i], "-s") || !strcmp(argv[i], "--seed")
             ){
                if(!argv[i+1] || !isValidComplex(argv[i+1])){
                        printf("Error: valor de seed ingresado no
                             valido\n");
                        return -1;
                } else {
                        char *copy = strdup(argv[i+1]);
                        if(copy == NULL){
                                exitCode = -1;
                                perror("Error impreso por perror");
                        }
                        int sign = 1;
                        if(copy[0] == '-') sign = -1;

                        pSeparator = strtok(argv[i+1],delimitator);
                        if(pSeparator != NULL){
                                seed.x = sign * atof(pSeparator);
                                int len = strlen(pSeparator);
                                if(sign == -1) sign = copy[len + 1]
                                     == '-' ? -1 : 1;
                                else sign = copy[len] == '-' ? -1 :
                                    1;
                        } else {
                                exitCode = -1;
                                perror("Error impreso por perror");
                        }

                        pSeparator = strtok (NULL,delimitator);
                        if(pSeparator != NULL){
                        seed.y = sign * atof(pSeparator);
                        } else {
                        exitCode = -1;
```

```
                        perror("Error impreso por perror");
                        }
                        free(copy);
                        i++;
                }
        }

        else if (!strcmp(argv[i], "-o") || !strcmp(argv[i], "--output
            ")){
                /* open output file */
                if(!argv[i+1]){
                        printf("Error: debe ingresar un archivo de
                            salida\n");
                        return -1;
                } else if (!strcmp(argv[i+1], "-")){
                        i++;
                        continue;
                }
                else {
                        image = fopen(argv[i+1], "w");
                        if (image == NULL) {
                                if(fprintf(stderr, "No se puede abrir
                                    el archivo file %s!\n", argv[i
                                    +1]) < 0){
                exitCode = -1;
                perror("Error impreso por perror");
                                }
                                perror("Error impreso por perror");
                                return -1;
                        }
                        i++;
                }
        } else {
                if(fprintf(stderr, "Error: opcion invalida\n") < 0){
        exitCode = -1;
        perror("Error impreso por perror");
                }
                perror("Error impreso por perror");
                return -1;
        }
}

if(exitCode == 0) {
        exitCode = processImage(resWidth,resHeight,pixelPos,seed,
            width,height,image,pasoN);
}

if(exitCode == EOF){
perror("Error impreso por perror despues de procesar la imagen");
}

return exitCode;
}
```

5. Pruebas

Para las pruebas compilamos el programa con gcc de la siguiente manera:

```
$gcc main.c -o tp0
```

Luego corremos el archivo **test.sh**. Ya que las pruebas son sobre las imágenes, las vamos a realizar a ojo comparandolas con las del enunciado y con las obtenidas en un generador online (`http://usefuljs.net/fractals/`).

Cabe destacar que las imagenes del generador tienen mayor rango dinamico que las del enunciado y nosotros decidimos generarlas como en éste último.

Las imagenes obtenidas por nuestro trabajo se encuentran también en formato PNG en la subcarpeta *imagenes*. A su vez las imagenes del generador online se encuentran en *Casos de prueba*.

5.1. Caso con los valores por defecto

Se obtiene una imagen como la primera figura del enunciado:

```
$./tp0 -o uno.pgm
```

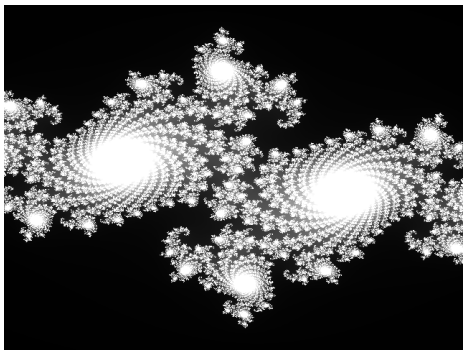

Figura 1:

5.2. Caso de imagen con zoom y otro centro

Se obtiene una imagen como la segunda figura del enunciado:

```
$ ./tp0 -c 0.282-0.007i -w 0.005 -H 0.005 -o dos.pgm
```

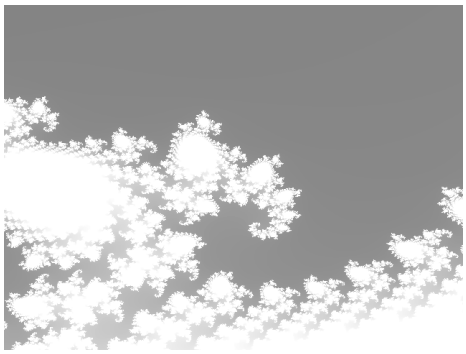

Figura 2:

5.3. Caso de imagen con ancho 1 y centro 1

Se obtiene una imagen como la primera del enunciado pero con un zoom x2 aplicado. Realizamos esta prueba por ser un caso muy facil de reproducir y comprarar con otro generador de Julia Sets.

```
$ ./tp0 -w 1 -H 1 -o tres.pgm
```

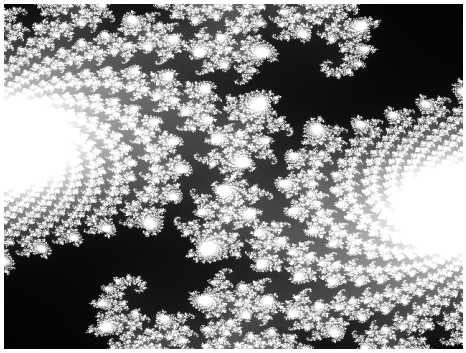

Figura 3:

5.4. Caso de imagen muy chica

Imprimimos una imagen de 8x6 para que se puedan notar claramente los pixeles en la pantalla al hacer zoom con GIMP.

```
$ ./tp0 -r 8x6 -o cuatro.pgm
```

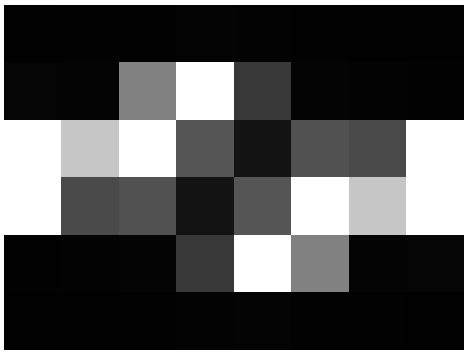

Figura 4:

5.5. Caso de imagen con otra semilla

Esta imagen usa una semilla con sus dos componentes negativas y la imaginaria mucho mas grande que la real. Esto es para comprobar que el centrado funcione correctamente, mas alla del caso del enunciado.

```
$ ./tp0 -s -0.157-1.041i -o cinco.pgm
```

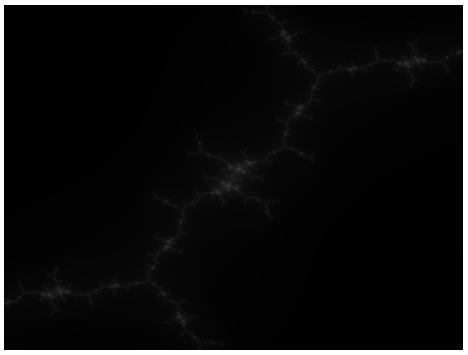

Figura 5:

5.6. Caso de imagen muy angosta

En este caso cambiamos la resolucion para obtener una imagen de un pixel de alto y 800 de ancho.
Esto se realizo para verificar que no ocurren problemas de dibujado en casos extremos. Es difícil observarla en el informe por lo que decidimos no colocarla. Sin embargo el archivo se encuentra en la misma carpeta del TP con el nombre *seis.pgm*.

```
$ ./tp0 -r 800x1 -o seis.pgm
```

6. Código S

En esta sección colocaremos el código assembly MIPS generado por NetBSD. El código se generó con la siguiente comando:

```
$ gcc -std=gnu99 -Wall -O0 -S -mrnames main.c
```

```
        .file   1 "main.c"
        .section .mdebug.abi32
        .previous
        .abicalls
        .text
        .align  2
        .globl  addComplexNumbers
        .ent    addComplexNumbers
addComplexNumbers:
        .frame  $fp,32,$ra              # vars= 16, regs= 2/0, args= 0, extra
            = 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,32
        .cprestore 0
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        move    $v1,$a0
        sw      $a2,40($fp)
        sw      $a3,44($fp)
        l.d     $f2,40($fp)
        l.d     $f0,56($fp)
        add.d   $f0,$f2,$f0
        s.d     $f0,8($fp)
        l.d     $f2,48($fp)
        l.d     $f0,64($fp)
        add.d   $f0,$f2,$f0
        s.d     $f0,16($fp)
        lw      $v0,8($fp)
        sw      $v0,0($v1)
        lw      $v0,12($fp)
        sw      $v0,4($v1)
        lw      $v0,16($fp)
        sw      $v0,8($v1)
        lw      $v0,20($fp)
        sw      $v0,12($v1)
        move    $v0,$v1
        move    $sp,$fp
        lw      $fp,28($sp)
        addu    $sp,$sp,32
        j       $ra
        .end    addComplexNumbers
        .size   addComplexNumbers, .-addComplexNumbers
```

```
        .align  2
        .globl  sqrComplex
        .ent    sqrComplex
sqrComplex:
        .frame  $fp,32,$ra              # vars= 16, regs= 2/0, args= 0, extra
            = 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,32
        .cprestore 0
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        move    $v1,$a0
        sw      $a2,40($fp)
        sw      $a3,44($fp)
        l.d     $f2,40($fp)
        l.d     $f0,40($fp)
        mul.d   $f4,$f2,$f0
        l.d     $f2,48($fp)
        l.d     $f0,48($fp)
        mul.d   $f0,$f2,$f0
        sub.d   $f0,$f4,$f0
        s.d     $f0,8($fp)
        l.d     $f0,40($fp)
        add.d   $f2,$f0,$f0
        l.d     $f0,48($fp)
        mul.d   $f0,$f2,$f0
        s.d     $f0,16($fp)
        lw      $v0,8($fp)
        sw      $v0,0($v1)
        lw      $v0,12($fp)
        sw      $v0,4($v1)
        lw      $v0,16($fp)
        sw      $v0,8($v1)
        lw      $v0,20($fp)
        sw      $v0,12($v1)
        move    $v0,$v1
        move    $sp,$fp
        lw      $fp,28($sp)
        addu    $sp,$sp,32
        j       $ra
        .end    sqrComplex
        .size   sqrComplex, .-sqrComplex
        .align  2
        .globl  absComplex
        .ent    absComplex
absComplex:
        .frame  $fp,40,$ra              # vars= 0, regs= 3/0, args= 16, extra
            = 8
        .mask   0xd0000000,-8
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,40
        .cprestore 16
        sw      $ra,32($sp)
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,40($fp)
        sw      $a1,44($fp)
        sw      $a2,48($fp)
        sw      $a3,52($fp)
        l.d     $f2,40($fp)
        l.d     $f0,40($fp)
        mul.d   $f4,$f2,$f0
        l.d     $f2,48($fp)
        l.d     $f0,48($fp)
        mul.d   $f0,$f2,$f0
```

```
        add.d   $f0,$f4,$f0
        mov.d   $f12,$f0
        la      $t9,sqrt
        jal     $ra,$t9
        move    $sp,$fp
        lw      $ra,32($sp)
        lw      $fp,28($sp)
        addu    $sp,$sp,40
        j       $ra
        .end    absComplex
        .size   absComplex, .-absComplex
        .rdata
        .align  2
$LC0:
        .ascii  "P2 \n\000"
        .align  2
$LC1:
        .ascii  "Error impreso por perror\000"
        .align  2
$LC2:
        .ascii  "%d %d \n\000"
        .align  2
$LC3:
        .ascii  "%d \n\000"
        .align  2
$LC5:
        .ascii  "%3d \000"
        .align  2
$LC6:
        .ascii  "\n\000"
        .align  3
$LC4:
        .word   0
        .word   1073741824
        .text
        .align  2
        .globl  processImage
        .ent    processImage
processImage:
        .frame  $fp,136,$ra             # vars= 72, regs= 4/0, args= 40,
            extra= 8
        .mask   0xd0010000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,136
        .cprestore 40
        sw      $ra,132($sp)
        sw      $fp,128($sp)
        sw      $gp,124($sp)
        sw      $s0,120($sp)
        move    $fp,$sp
        sw      $a0,136($fp)
        sw      $a1,140($fp)
        sw      $a2,144($fp)
        sw      $a3,148($fp)
        lw      $a0,192($fp)
        la      $a1,$LC0
        la      $t9,fprintf
        jal     $ra,$t9
        bgez    $v0,$L21
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,112($fp)
        b       $L20
$L21:
        lw      $a0,192($fp)
        la      $a1,$LC2
        lw      $a2,136($fp)
        lw      $a3,140($fp)
        la      $t9,fprintf
```

```
        jal     $ra,$t9
        bgez    $v0,$L22
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,112($fp)
        b       $L20
$L22:
        lw      $a0,192($fp)
        la      $a1,$LC3
        lw      $a2,196($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        bgez    $v0,$L23
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,112($fp)
        b       $L20
$L23:
        sw      $zero,52($fp)
$L24:
        lw      $v0,52($fp)
        lw      $v1,140($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L27
        b       $L25
$L27:
        sw      $zero,48($fp)
$L28:
        lw      $v0,48($fp)
        lw      $v1,136($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L31
        b       $L29
$L31:
        l.d     $f2,176($fp)
        l.d     $f0,$LC4
        div.d   $f2,$f2,$f0
        l.d     $f0,144($fp)
        sub.d   $f4,$f0,$f2
        l.s     $f0,136($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,176($fp)
        div.d   $f2,$f0,$f2
        l.d     $f0,$LC4
        div.d   $f0,$f2,$f0
        add.d   $f4,$f4,$f0
        l.s     $f0,136($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,176($fp)
        div.d   $f2,$f0,$f2
        l.s     $f0,48($fp)
        cvt.d.w $f0,$f0
        mul.d   $f0,$f2,$f0
        add.d   $f0,$f4,$f0
        s.d     $f0,80($fp)
        l.d     $f2,184($fp)
        l.d     $f0,$LC4
        div.d   $f2,$f2,$f0
        l.d     $f0,152($fp)
        add.d   $f4,$f2,$f0
        l.s     $f0,140($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,184($fp)
        div.d   $f2,$f0,$f2
        l.d     $f0,$LC4
        div.d   $f0,$f2,$f0
        sub.d   $f4,$f4,$f0
        l.s     $f0,140($fp)
        cvt.d.w $f2,$f0
        l.d     $f0,184($fp)
```

```
        div.d   $f2,$f0,$f2
        l.s     $f0,52($fp)
        cvt.d.w $f0,$f0
        mul.d   $f0,$f2,$f0
        sub.d   $f0,$f4,$f0
        s.d     $f0,88($fp)
        sw      $zero,64($fp)
        sw      $zero,68($fp)
        sw      $zero,72($fp)
        sw      $zero,76($fp)
        sw      $zero,56($fp)
$L32:
        lw      $v0,196($fp)
        addu    $v1,$v0,-1
        lw      $v0,56($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L35
        b       $L33
$L35:
        addu    $s0,$fp,64
        addu    $v1,$fp,96
        lw      $v0,88($fp)
        sw      $v0,16($sp)
        lw      $v0,92($fp)
        sw      $v0,20($sp)
        lw      $a2,80($fp)
        lw      $a3,84($fp)
        move    $a0,$v1
        la      $t9,sqrComplex
        jal     $ra,$t9
        lw      $v0,160($fp)
        sw      $v0,24($sp)
        lw      $v0,164($fp)
        sw      $v0,28($sp)
        lw      $v0,168($fp)
        sw      $v0,32($sp)
        lw      $v0,172($fp)
        sw      $v0,36($sp)
        lw      $v0,104($fp)
        sw      $v0,16($sp)
        lw      $v0,108($fp)
        sw      $v0,20($sp)
        lw      $a2,96($fp)
        lw      $a3,100($fp)
        move    $a0,$s0
        la      $t9,addComplexNumbers
        jal     $ra,$t9
        lw      $v0,64($fp)
        sw      $v0,80($fp)
        lw      $v0,68($fp)
        sw      $v0,84($fp)
        lw      $v0,72($fp)
        sw      $v0,88($fp)
        lw      $v0,76($fp)
        sw      $v0,92($fp)
        lw      $a0,64($fp)
        lw      $a1,68($fp)
        lw      $a2,72($fp)
        lw      $a3,76($fp)
        la      $t9,absComplex
        jal     $ra,$t9
        mov.d   $f2,$f0
        l.d     $f0,$LC4
        c.lt.d  $f0,$f2
        bc1t    $L33
        lw      $v0,56($fp)
        addu    $v0,$v0,1
        sw      $v0,56($fp)
        lw      $v0,56($fp)
        addu    $v0,$v0,1
        sw      $v0,56($fp)
        b       $L32
$L33:
        lw      $a0,192($fp)
```

```
        la      $a1,$LC5
        lw      $a2,56($fp)
        la      $t9,fprintf
        jal     $ra,$t9
        bgez    $v0,$L30
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,112($fp)
        b       $L20
$L30:
        lw      $v0,48($fp)
        addu    $v0,$v0,1
        sw      $v0,48($fp)
        b       $L28
$L29:
        lw      $a0,192($fp)
        la      $a1,$LC6
        la      $t9,fprintf
        jal     $ra,$t9
        bgez    $v0,$L26
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,112($fp)
        b       $L20
$L26:
        lw      $v0,52($fp)
        addu    $v0,$v0,1
        sw      $v0,52($fp)
        b       $L24
$L25:
        lw      $a0,192($fp)
        la      $t9,fclose
        jal     $ra,$t9
        sw      $v0,112($fp)
$L20:
        lw      $v0,112($fp)
        move    $sp,$fp
        lw      $ra,132($sp)
        lw      $fp,128($sp)
        lw      $s0,120($sp)
        addu    $sp,$sp,136
        j       $ra
        .end    processImage
        .size   processImage, .-processImage
        .align  2
        .globl  isValidNumber
        .ent    isValidNumber
isValidNumber:
        .frame  $fp,24,$ra              # vars= 8, regs= 2/0, args= 0, extra=
            8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,24
        .cprestore 0
        sw      $fp,20($sp)
        sw      $gp,16($sp)
        move    $fp,$sp
        sw      $a0,24($fp)
        sw      $zero,8($fp)
$L41:
        lw      $v1,24($fp)
        lw      $v0,8($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
        bne     $v0,$zero,$L44
        b       $L42
$L44:
```

```
        lw      $v1,24($fp)
        lw      $v0,8($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
        slt     $v0,$v0,48
        bne     $v0,$zero,$L46
        lw      $v1,24($fp)
        lw      $v0,8($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
        slt     $v0,$v0,58
        beq     $v0,$zero,$L46
        b       $L43
$L46:
        lw      $v1,24($fp)
        lw      $v0,8($fp)
        addu    $v0,$v1,$v0
        lb      $v1,0($v0)
        li      $v0,46                  # 0x2e
        bne     $v1,$v0,$L47
        lw      $v0,8($fp)
        beq     $v0,$zero,$L49
        lw      $v1,24($fp)
        lw      $v0,8($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v0,0($v0)
        bne     $v0,$zero,$L43
$L49:
        sw      $zero,12($fp)
        b       $L40
$L47:
        sw      $zero,12($fp)
        b       $L40
$L43:
        lw      $v0,8($fp)
        addu    $v0,$v0,1
        sw      $v0,8($fp)
        b       $L41
$L42:
        li      $v0,1                   # 0x1
        sw      $v0,12($fp)
$L40:
        lw      $v0,12($fp)
        move    $sp,$fp
        lw      $fp,20($sp)
        addu    $sp,$sp,24
        j       $ra
        .end    isValidNumber
        .size   isValidNumber, .-isValidNumber
        .align  2
        .globl  isValidRes
        .ent    isValidRes
isValidRes:
        .frame  $fp,32,$ra              # vars= 16, regs= 2/0, args= 0, extra
            = 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,32
        .cprestore 0
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,32($fp)
        sw      $zero,8($fp)
        sw      $zero,12($fp)
$L51:
        lw      $v1,32($fp)
        lw      $v0,12($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
```

```
        bne     $v0,$zero,$L54
        b       $L52
$L54:
        lw      $v1,32($fp)
        lw      $v0,12($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
        slt     $v0,$v0,48
        bne     $v0,$zero,$L56
        lw      $v1,32($fp)
        lw      $v0,12($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
        slt     $v0,$v0,58
        beq     $v0,$zero,$L56
        b       $L53
$L56:
        lw      $v1,32($fp)
        lw      $v0,12($fp)
        addu    $v0,$v1,$v0
        lb      $v1,0($v0)
        li      $v0,120                 # 0x78
        bne     $v1,$v0,$L57
        lw      $v0,12($fp)
        beq     $v0,$zero,$L59
        lw      $v1,32($fp)
        lw      $v0,12($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v0,0($v0)
        bne     $v0,$zero,$L58
$L59:
        sw      $zero,16($fp)
        b       $L50
$L58:
        li      $v0,1                   # 0x1
        sw      $v0,8($fp)
        b       $L53
$L57:
        sw      $zero,16($fp)
        b       $L50
$L53:
        lw      $v0,12($fp)
        addu    $v0,$v0,1
        sw      $v0,12($fp)
        b       $L51
$L52:
        lw      $v0,8($fp)
        sw      $v0,16($fp)
$L50:
        lw      $v0,16($fp)
        move    $sp,$fp
        lw      $fp,28($sp)
        addu    $sp,$sp,32
        j       $ra
        .end    isValidRes
        .size   isValidRes, .-isValidRes
        .align  2
        .globl  isValidComplex
        .ent    isValidComplex
isValidComplex:
        .frame  $fp,32,$ra              # vars= 16, regs= 2/0, args= 0, extra
            = 8
        .mask   0x50000000,-4
        .fmask  0x00000000,0
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,32
        .cprestore 0
        sw      $fp,28($sp)
        sw      $gp,24($sp)
        move    $fp,$sp
        sw      $a0,32($fp)
```

```
        sw      $zero,8($fp)
        sw      $zero,12($fp)
        sw      $zero,16($fp)
$L61:
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
        bne     $v0,$zero,$L64
        b       $L62
$L64:
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
        slt     $v0,$v0,48
        bne     $v0,$zero,$L66
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        lb      $v0,0($v0)
        slt     $v0,$v0,58
        beq     $v0,$zero,$L66
        b       $L63
$L66:
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        lb      $v1,0($v0)
        li      $v0,46                  # 0x2e
        bne     $v1,$v0,$L67
        lw      $v0,16($fp)
        beq     $v0,$zero,$L69
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,105                 # 0x69
        beq     $v1,$v0,$L69
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,43                  # 0x2b
        beq     $v1,$v0,$L69
        b       $L63
$L69:
        sw      $zero,20($fp)
        b       $L60
$L67:
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        lb      $v1,0($v0)
        li      $v0,105                 # 0x69
        bne     $v1,$v0,$L71
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v0,0($v0)
        beq     $v0,$zero,$L72
        sw      $zero,20($fp)
        b       $L60
$L72:
        lw      $v0,12($fp)
        sw      $v0,8($fp)
        b       $L63
$L71:
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
```

```
        lb      $v1,0($v0)
        li      $v0,43                  # 0x2b
        bne     $v1,$v0,$L74
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,45                  # 0x2d
        beq     $v1,$v0,$L76
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,105                 # 0x69
        beq     $v1,$v0,$L76
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,46                  # 0x2e
        beq     $v1,$v0,$L76
        lw      $v0,16($fp)
        bne     $v0,$zero,$L75
$L76:
        sw      $zero,20($fp)
        b       $L60
$L75:
        li      $v0,1                   # 0x1
        sw      $v0,12($fp)
        b       $L63
$L74:
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        lb      $v1,0($v0)
        li      $v0,45                  # 0x2d
        bne     $v1,$v0,$L70
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,43                  # 0x2b
        beq     $v1,$v0,$L80
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,46                  # 0x2e
        beq     $v1,$v0,$L80
        lw      $v1,32($fp)
        lw      $v0,16($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,105                 # 0x69
        beq     $v1,$v0,$L80
        b       $L79
$L80:
        sw      $zero,20($fp)
        b       $L60
$L79:
        lw      $v0,16($fp)
        beq     $v0,$zero,$L63
        li      $v0,1                   # 0x1
        sw      $v0,12($fp)
        b       $L63
$L70:
        sw      $zero,20($fp)
        b       $L60
```

```
$L63:
        lw      $v0,16($fp)
        addu    $v0,$v0,1
        sw      $v0,16($fp)
        b       $L61
$L62:
        lw      $v0,8($fp)
        sw      $v0,20($fp)
$L60:
        lw      $v0,20($fp)
        move    $sp,$fp
        lw      $fp,28($sp)
        addu    $sp,$sp,32
        j       $ra
        .end    isValidComplex
        .size   isValidComplex, .-isValidComplex
        .rdata
        .align  2
$LC7:
        .ascii  "x\000"
        .align  2
$LC8:
        .ascii  "+-i\000"
        .align  2
$LC12:
        .ascii  "-V\000"
        .align  2
$LC13:
        .ascii  "--version\000"
        .align  2
$LC14:
        .ascii  "TP0 Organizacion de Computadoras version \"1.0.0\"      "
        .ascii  "        \n\n"
        .ascii  "Integrantes:\n"
        .ascii  " Fabrizio Cozza\n"
        .ascii  " Kevin Cajachu\303\241n\n"
        .ascii  " Luciano Giannotti\n\000"
        .align  2
$LC15:
        .ascii  "-h\000"
        .align  2
$LC16:
        .ascii  "--help\000"
        .align  2
$LC17:
        .ascii  "Uso:\n"
        .ascii  "  tp0 -h\n"
        .ascii  "  tp0 -V\n"
        .ascii  "  tp0 [options]\n"
        .ascii  "Opciones:\n"
        .ascii  "  -V, --version     Version del programa.\n"
        .ascii  "  -h, --help        Informacion acerca de los comandos.\n"
        .ascii  "  -r, --resolution  Cambiar resolucion de la imagen.\n"
        .ascii  "  -c, --center      Coordenadas correspondientes al punt"
        .ascii  "o central.\n"
        .ascii  "  -w, --width       Especifica el ancho de la region del"
        .ascii  " plano complejo por dibujar.\n"
        .ascii  "  -H, --height      Especifica el alto de la region del "
        .ascii  "plano complejo por dibujar.\n"
        .ascii  "  -s, --seed        Configurar el valor complejo de la s"
        .ascii  "emilla usada para generar el fractal.\n"
        .ascii  "  -o, --output      Colocar la imagen de salida.\n"
        .ascii  "Ejemplos:\n"
        .ascii  "  tp0 -o uno.pgm\n\000"
        .align  2
$LC18:
        .ascii  "-r\000"
        .align  2
$LC19:
        .ascii  "--resolution\000"
        .align  2
$LC20:
        .ascii  "Error: valor de resolucion ingresado no valido\n\000"
        .align  2
```

```
$LC21:
        .ascii  "-c\000"
        .align  2
$LC22:
        .ascii  "--center\000"
        .align  2
$LC23:
        .ascii  "Error: valor de centro ingresado no valido\n\000"
        .align  2
$LC24:
        .ascii  "-w\000"
        .align  2
$LC25:
        .ascii  "--width\000"
        .align  2
$LC26:
        .ascii  "Error: valor de ancho ingresado no valido\n\000"
        .align  2
$LC27:
        .ascii  "-H\000"
        .align  2
$LC28:
        .ascii  "--height\000"
        .align  2
$LC29:
        .ascii  "Error: valor de altura ingresado no valido\n\000"
        .align  2
$LC30:
        .ascii  "-s\000"
        .align  2
$LC31:
        .ascii  "--seed\000"
        .align  2
$LC32:
        .ascii  "Error: valor de seed ingresado no valido\n\000"
        .align  2
$LC33:
        .ascii  "-o\000"
        .align  2
$LC34:
        .ascii  "--output\000"
        .align  2
$LC35:
        .ascii  "Error: debe ingresar un archivo de salida\n\000"
        .align  2
$LC36:
        .ascii  "-\000"
        .align  2
$LC37:
        .ascii  "w\000"
        .align  2
$LC38:
        .ascii  "No se puede abrir el archivo file %s!\n\000"
        .align  2
$LC39:
        .ascii  "Error: opcion invalida\n\000"
        .align  2
$LC40:
        .ascii  "Error impreso por perror despues de procesar la imagen\000"
        .align  3
$LC9:
        .word   0
        .word   1073741824
        .align  3
$LC10:
        .word   138464867
        .word   -1075363142
        .align  3
$LC11:
        .word   351303579
        .word   1070083444
        .text
        .align  2
        .globl  main
```

```
        .ent    main
main:
        .frame  $fp,216,$ra             # vars= 120, regs= 3/1, args= 64,
            extra= 8
        .mask   0xd0000000,-16
        .fmask  0x00300000,-8
        .set    noreorder
        .cpload $t9
        .set    reorder
        subu    $sp,$sp,216
        .cprestore 64
        sw      $ra,200($sp)
        sw      $fp,196($sp)
        sw      $gp,192($sp)
        s.d     $f20,208($sp)
        move    $fp,$sp
        sw      $a0,216($fp)
        sw      $a1,220($fp)
        sw      $zero,72($fp)
        li      $v0,500                 # 0x1f4
        sw      $v0,76($fp)
        la      $v0,__sF+88
        sw      $v0,136($fp)
        la      $v0,$LC7
        sw      $v0,140($fp)
        la      $v0,$LC8
        sw      $v0,144($fp)
        li      $v0,640                 # 0x280
        sw      $v0,80($fp)
        li      $v0,480                 # 0x1e0
        sw      $v0,84($fp)
        sw      $zero,88($fp)
        sw      $zero,92($fp)
        sw      $zero,96($fp)
        sw      $zero,100($fp)
        l.d     $f0,$LC9
        s.d     $f0,104($fp)
        l.d     $f0,$LC9
        s.d     $f0,112($fp)
        l.d     $f0,$LC10
        s.d     $f0,120($fp)
        l.d     $f0,$LC11
        s.d     $f0,128($fp)
        li      $v0,1                   # 0x1
        sw      $v0,152($fp)
$L83:
        lw      $v0,152($fp)
        lw      $v1,216($fp)
        slt     $v0,$v0,$v1
        bne     $v0,$zero,$L86
        b       $L84
$L86:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC12
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L88
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC13
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L87
$L88:
        la      $a0,$LC14
        la      $t9,printf
        jal     $ra,$t9
```

```
        sw      $zero,168($fp)
        b       $L82
$L87:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC15
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L91
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC16
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L90
$L91:
        la      $a0,$LC17
        la      $t9,printf
        jal     $ra,$t9
        sw      $zero,168($fp)
        b       $L82
$L90:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC18
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L94
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC19
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L93
$L94:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        beq     $v0,$zero,$L96
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,isValidRes
        jal     $ra,$t9
        bne     $v0,$zero,$L95
$L96:
        la      $a0,$LC20
        la      $t9,printf
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,168($fp)
        b       $L82
$L95:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
```

```
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        lw      $a1,140($fp)
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,148($fp)
        lw      $v0,148($fp)
        beq     $v0,$zero,$L98
        lw      $a0,148($fp)
        la      $t9,atof
        jal     $ra,$t9
        trunc.w.d $f0,$f0,$v0
        s.s     $f0,80($fp)
        b       $L99
$L98:
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L99:
        move    $a0,$zero
        lw      $a1,140($fp)
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,148($fp)
        lw      $v0,148($fp)
        beq     $v0,$zero,$L100
        lw      $a0,148($fp)
        la      $t9,atof
        jal     $ra,$t9
        trunc.w.d $f0,$f0,$v0
        s.s     $f0,84($fp)
        b       $L101
$L100:
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L101:
        lw      $v0,152($fp)
        addu    $v0,$v0,1
        sw      $v0,152($fp)
        b       $L85
$L93:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC21
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L104
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC22
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L103
$L104:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        beq     $v0,$zero,$L106
        lw      $v0,152($fp)
```

```
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,isValidComplex
        jal     $ra,$t9
        bne     $v0,$zero,$L105
$L106:
        la      $a0,$LC23
        la      $t9,printf
        jal     $ra,$t9
        li      $v1,-1                  # 0xffffffffffffffff
        sw      $v1,168($fp)
        b       $L82
$L105:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,strdup
        jal     $ra,$t9
        sw      $v0,156($fp)
        lw      $v0,156($fp)
        bne     $v0,$zero,$L108
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L108:
        li      $v0,1                   # 0x1
        sw      $v0,160($fp)
        lw      $v0,156($fp)
        lb      $v1,0($v0)
        li      $v0,45                  # 0x2d
        bne     $v1,$v0,$L109
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,160($fp)
$L109:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        lw      $a1,144($fp)
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,148($fp)
        lw      $v0,148($fp)
        beq     $v0,$zero,$L110
        l.s     $f0,160($fp)
        cvt.d.w $f20,$f0
        lw      $a0,148($fp)
        la      $t9,atof
        jal     $ra,$t9
        mul.d   $f0,$f20,$f0
        s.d     $f0,88($fp)
        lw      $a0,148($fp)
        la      $t9,strlen
        jal     $ra,$t9
        sw      $v0,164($fp)
        lw      $v1,160($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L111
        lw      $v1,156($fp)
        lw      $v0,164($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,45                  # 0x2d
```

```
        bne     $v1,$v0,$L112
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,172($fp)
        b       $L113
$L112:
        li      $v1,1                   # 0x1
        sw      $v1,172($fp)
$L113:
        lw      $v0,172($fp)
        sw      $v0,160($fp)
        b       $L117
$L111:
        lw      $v1,156($fp)
        lw      $v0,164($fp)
        addu    $v0,$v1,$v0
        lb      $v1,0($v0)
        li      $v0,45                  # 0x2d
        bne     $v1,$v0,$L115
        li      $v1,-1                  # 0xffffffffffffffff
        sw      $v1,176($fp)
        b       $L116
$L115:
        li      $v0,1                   # 0x1
        sw      $v0,176($fp)
$L116:
        lw      $v1,176($fp)
        sw      $v1,160($fp)
        b       $L117
$L110:
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L117:
        move    $a0,$zero
        lw      $a1,144($fp)
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,148($fp)
        lw      $v0,148($fp)
        beq     $v0,$zero,$L118
        l.s     $f0,160($fp)
        cvt.d.w $f20,$f0
        lw      $a0,148($fp)
        la      $t9,atof
        jal     $ra,$t9
        mul.d   $f0,$f20,$f0
        s.d     $f0,96($fp)
        b       $L119
$L118:
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L119:
        lw      $a0,156($fp)
        la      $t9,free
        jal     $ra,$t9
        lw      $v0,152($fp)
        addu    $v0,$v0,1
        sw      $v0,152($fp)
        b       $L85
$L103:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC24
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L122
```

```
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC25
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L121
$L122:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        beq     $v0,$zero,$L124
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,isValidNumber
        jal     $ra,$t9
        bne     $v0,$zero,$L123
$L124:
        la      $a0,$LC26
        la      $t9,printf
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,168($fp)
        b       $L82
$L123:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,atof
        jal     $ra,$t9
        s.d     $f0,104($fp)
        lw      $v0,152($fp)
        addu    $v0,$v0,1
        sw      $v0,152($fp)
        b       $L85
$L121:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC27
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L128
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC28
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L127
$L128:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        beq     $v0,$zero,$L130
```

```
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,isValidNumber
        jal     $ra,$t9
        bne     $v0,$zero,$L129
$L130:
        la      $a0,$LC29
        la      $t9,printf
        jal     $ra,$t9
        li      $v1,-1                  # 0xffffffffffffffff
        sw      $v1,168($fp)
        b       $L82
$L129:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,atof
        jal     $ra,$t9
        s.d     $f0,112($fp)
        lw      $v0,152($fp)
        addu    $v0,$v0,1
        sw      $v0,152($fp)
        b       $L85
$L127:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC30
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L134
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC31
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L133
$L134:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
        beq     $v0,$zero,$L136
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,isValidComplex
        jal     $ra,$t9
        bne     $v0,$zero,$L135
$L136:
        la      $a0,$LC32
        la      $t9,printf
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,168($fp)
        b       $L82
$L135:
```

```
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $t9,strdup
        jal     $ra,$t9
        sw      $v0,164($fp)
        lw      $v0,164($fp)
        bne     $v0,$zero,$L138
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L138:
        li      $v0,1                   # 0x1
        sw      $v0,160($fp)
        lw      $v0,164($fp)
        lb      $v1,0($v0)
        li      $v0,45                  # 0x2d
        bne     $v1,$v0,$L139
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,160($fp)
$L139:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        lw      $a1,144($fp)
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,148($fp)
        lw      $v0,148($fp)
        beq     $v0,$zero,$L140
        l.s     $f0,160($fp)
        cvt.d.w $f20,$f0
        lw      $a0,148($fp)
        la      $t9,atof
        jal     $ra,$t9
        mul.d   $f0,$f20,$f0
        s.d     $f0,120($fp)
        lw      $a0,148($fp)
        la      $t9,strlen
        jal     $ra,$t9
        sw      $v0,156($fp)
        lw      $v1,160($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L141
        lw      $v1,164($fp)
        lw      $v0,156($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,1
        lb      $v1,0($v0)
        li      $v0,45                  # 0x2d
        bne     $v1,$v0,$L142
        li      $v1,-1                  # 0xffffffffffffffff
        sw      $v1,180($fp)
        b       $L143
$L142:
        li      $v0,1                   # 0x1
        sw      $v0,180($fp)
$L143:
        lw      $v1,180($fp)
        sw      $v1,160($fp)
        b       $L147
$L141:
        lw      $v1,164($fp)
        lw      $v0,156($fp)
        addu    $v0,$v1,$v0
        lb      $v1,0($v0)
```

```
        li      $v0,45                  # 0x2d
        bne     $v1,$v0,$L145
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,184($fp)
        b       $L146
$L145:
        li      $v1,1                   # 0x1
        sw      $v1,184($fp)
$L146:
        lw      $v0,184($fp)
        sw      $v0,160($fp)
        b       $L147
$L140:
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L147:
        move    $a0,$zero
        lw      $a1,144($fp)
        la      $t9,strtok
        jal     $ra,$t9
        sw      $v0,148($fp)
        lw      $v0,148($fp)
        beq     $v0,$zero,$L148
        l.s     $f0,160($fp)
        cvt.d.w $f20,$f0
        lw      $a0,148($fp)
        la      $t9,atof
        jal     $ra,$t9
        mul.d   $f0,$f20,$f0
        s.d     $f0,128($fp)
        b       $L149
$L148:
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L149:
        lw      $a0,164($fp)
        la      $t9,free
        jal     $ra,$t9
        lw      $v0,152($fp)
        addu    $v0,$v0,1
        sw      $v0,152($fp)
        b       $L85
$L133:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC33
        la      $t9,strcmp
        jal     $ra,$t9
        beq     $v0,$zero,$L152
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        lw      $a0,0($v0)
        la      $a1,$LC34
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L151
$L152:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $v0,0($v0)
```

```
        bne     $v0,$zero,$L153
        la      $a0,$LC35
        la      $t9,printf
        jal     $ra,$t9
        li      $v1,-1                  # 0xffffffffffffffff
        sw      $v1,168($fp)
        b       $L82
$L153:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC36
        la      $t9,strcmp
        jal     $ra,$t9
        bne     $v0,$zero,$L155
        lw      $v0,152($fp)
        addu    $v0,$v0,1
        sw      $v0,152($fp)
        b       $L85
$L155:
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        lw      $a0,0($v0)
        la      $a1,$LC37
        la      $t9,fopen
        jal     $ra,$t9
        sw      $v0,136($fp)
        lw      $v0,136($fp)
        bne     $v0,$zero,$L157
        lw      $v0,152($fp)
        sll     $v1,$v0,2
        lw      $v0,220($fp)
        addu    $v0,$v1,$v0
        addu    $v0,$v0,4
        la      $a0,__sF+176
        la      $a1,$LC38
        lw      $a2,0($v0)
        la      $t9,fprintf
        jal     $ra,$t9
        bgez    $v0,$L158
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
$L158:
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,168($fp)
        b       $L82
$L157:
        lw      $v0,152($fp)
        addu    $v0,$v0,1
        sw      $v0,152($fp)
        b       $L85
$L151:
        la      $a0,__sF+176
        la      $a1,$LC39
        la      $t9,fprintf
        jal     $ra,$t9
        bgez    $v0,$L160
        li      $v0,-1                  # 0xffffffffffffffff
        sw      $v0,72($fp)
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
```

```
$L160:
        la      $a0,$LC1
        la      $t9,perror
        jal     $ra,$t9
        li      $v1,-1                  # 0xffffffffffffffff
        sw      $v1,168($fp)
        b       $L82
$L85:
        lw      $v0,152($fp)
        addu    $v0,$v0,1
        sw      $v0,152($fp)
        b       $L83
$L84:
        lw      $v0,72($fp)
        bne     $v0,$zero,$L161
        lw      $v0,120($fp)
        sw      $v0,24($sp)
        lw      $v0,124($fp)
        sw      $v0,28($sp)
        lw      $v0,128($fp)
        sw      $v0,32($sp)
        lw      $v0,132($fp)
        sw      $v0,36($sp)
        l.d     $f0,104($fp)
        s.d     $f0,40($sp)
        l.d     $f0,112($fp)
        s.d     $f0,48($sp)
        lw      $v0,136($fp)
        sw      $v0,56($sp)
        lw      $v0,76($fp)
        sw      $v0,60($sp)
        lw      $v0,96($fp)
        sw      $v0,16($sp)
        lw      $v0,100($fp)
        sw      $v0,20($sp)
        lw      $a2,88($fp)
        lw      $a3,92($fp)
        lw      $a0,80($fp)
        lw      $a1,84($fp)
        la      $t9,processImage
        jal     $ra,$t9
        sw      $v0,72($fp)
$L161:
        lw      $v1,72($fp)
        li      $v0,-1                  # 0xffffffffffffffff
        bne     $v1,$v0,$L162
        la      $a0,$LC40
        la      $t9,perror
        jal     $ra,$t9
$L162:
        lw      $v0,72($fp)
        sw      $v0,168($fp)
$L82:
        lw      $v0,168($fp)
        move    $sp,$fp
        lw      $ra,200($sp)
        lw      $fp,196($sp)
        l.d     $f20,208($sp)
        addu    $sp,$sp,216
        j       $ra
        .end    main
        .size   main, .-main
        .ident  "GCC: (GNU) 3.3.3 (NetBSD nb3 20040520)"
```

7. Bibliografía

1. GXemul.
   http://gavare.se/gxemul/.
2. The NetBSD project.
   http://www.netbsd.org/.
3. http://es.wikipedia.org/wiki/Conjunto_de_Julia (Wikipedia).
4. PGM format specification.
   http://netpbm.sourceforge.net/doc/pgm.html.
5. Generador de fractales.
   http://usefuljs.net/fractals/
6. GIMP.
   https://www.gimp.org/